AD-A061 812 TRACOR INC AUSTIN TEX F/G 9/2
DEVELOPMENT AID FOR AN EMULATING MICROPROCESSOR.(U)
JUN 78 R FOSDICK F33615-77-C-1115
UNCLASSIFIED AFAL-TR-78-77 NL

| OF |
AD
A061812

END
DATE
FILMED
2 79
DDC

AFAL-TR-78-77

LEVEL II

2

DEVELOPMENT AID FOR AN EMULATING MICROPROCESSOR

R. FOSDICK
TRACOR, INC.
AUSTIN, TEXAS 78721

JUNE 1978

TECHNICAL REPORT AFAL-TR-78-77
Final Report 14 February 1977 – 28 February 1978

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) **REPORT DOCUMENTATION PAGE**		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) AFAL-TR-78-77	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) DEVELOPMENT AID FOR AN EMULATING MICROPROCESSOR.		5. TYPE OF REPORT & PERIOD COVERED (9) Final Report. 14 FEB 1977 - 28 FEB 1978
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) R. Fosdick		8. CONTRACT OR GRANT NUMBER(s) (15) F33615-77-C-1115 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Tracor, Inc. Austin, TX 78721		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 6096 (17) 40-05
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (DHE) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE (11) June 1978
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 33p.		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

General Processor Unit (GPU)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is the final report of a program initiated to implement a complete emulator utilizing the GPU circuit. When it was discovered that the GPU was going to require some mask changes the effort was redirected to start the definition of the LSI Micro Controller Circuit and to start developing an LSI Simulator for systems interface verification. This report covers work performed to establish the initial requirements and characteristics of a high performance microprogram controller IC. Some key elements of the study were: op-code decomposition techniques of target machines, definition of the controller

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

352 100

LB

function, and architecture of target machines with regards to requirements for the controller IC. The information developed under this contract will be of direct use in a follow-on contract to design and develop the controller IC.

PREFACE

This report covers work in preparation to a follow-on contract for the design and development of a micro program controller IC in support of the General Processor Unit (GPU).

The work performed includes the detailed examination of the architectures of target machines, op-code decomposition, constraints placed upon the controller function, and the characteristics and requirements of an advanced micro program controller. The purpose of the preliminary investigation was to characterize the target machines and to correlate the architectural and op-code requirements placed upon a controller IC.

This report was prepared by Tracor Inc., Applied Technology Systems Division, Austin, Texas, under Air Force contract F33615-77-C-1115. The contract is titled "Development Aid for an Emulating Microprocessor". Amendment P00002 modified the contract to initiate the analysis to define a micro-controller circuit compatible with the GPU circuit family. The report describes work performed during the period of February 1977 through December 1977. The effort was sponsored by the Air Force Avionics Laboratory, Wright-Patterson AFB, OH. Captain Thomas Margraff (AFAL/DHE) was contract monitor.

The report was submitted January 1978.

Reviewed and Approved:

Approved: [illegible]
ROBERT FOSDICK
Program Director, Computer Devices
Teletypewriters and Digital Systems

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

SECTION I

CONTROLLER REQUIREMENTS STUDY

1. INTRODUCTION

The family of circuits the controller is to be a part of are fabricated with the CMOS/SOS process. These circuits have certain inherent features desirable to the military for the implementation of computers which include:

1) Low power
2) High performance
3) Good radiation tolerance
4) High tolerance to voltage variation
5) High noise immunity
6) Single voltage requirement
7) Good packing densities

As with any new circuit technology development there is not a large base of existing circuits available to supplement the new designs and it is undesirable to mix technologies as the advantages peculiar to each technology would be lost. Therefore, it is desirable to make each device in the circuit family as complete as possible, thus minimizing the need for additional special purpose circuits. The object of this study is to survey the control functions in military avionics processors and define the requirements for the controller. It is desirable that the device be capable of implementing various control type functions in different areas of a computer with the goal of minimizing the requirements for additional unique circuit developments.

2. PROCESSOR CHARACTERISTICS SURVEY

a. PDP-11 Characteristics. The PDP-11 family of processors start with a basic instruction set and a range of hardware performances most suited for this application. The PDP-11 is a 16-bit architecture with all instructions also being 16-bits. Following are a list of features in the PDP-11 family that relate directly to the control functions:

1) 16-bit word
2) Direct addressing of 32K 16-bit words
3) Word or byte processing
4) Stack processing
5) Direct Memory Access (DMA)
6) 8 internal, general - purpose registers
7) Vectored, priority interrupts
8) Single and double operand instructions
9) Multiply/divide and floating point options
10) Status register (1)

(1) PDP-11 Instruction Description. The PDP-11 instruction employs the use of many instruction formats with multiple field definitions to provide the memory addressing flexibility that other machines provide by using a 32-bit instruction. The basic instruction formats are shown in Figure 1. The operand derivations provided for both the source and destination fields represent a good set that contribute much to acceptance of this family. The operand derivation list is shown in Figure 2.

b. UYK-20, AYK-14 Characteristics. The AYK-14 is the Navy's airborne computer also referred to as the ISADC (Interim Standard Airborne Digital Computer). The AYK-14 emulates the UYK-20 instruction set and includes several additional instructions unique to the AYK-14. The UYK-20 has been the Navy's standard shipboard computer for the past several years and represents the nearest to a standard computer architecture for the Navy.

The UYK-20 and AYK-14 computers are 16-bit general purpose machines incorporating an extensive instruction set that has evolved from previous Navy computers. Following are a list of the functional features relating to the control requirements:

1) 8-bit, 16-bit, 32-bit operands
2) 16 general purpose registers
3) 2 program status registers
4) 16-bit and 32-bit instructions
5) Direct addressing by page
6) Relative addressing by page
7) Cascade indirect addressing
8) 3-level interrupt processing
9) MATHPAC option
10) Input-output controller
11) Indexing via general registers
12) 16-bit memory word
13) Real-time and monitor clocks
14) Direct Memory Access (DMA)

(1) UYK-20 Instruction Description. The UYK-20 has a 6-bit op-code field, a 2-bit format field, and two 4-bit register fields. The first register field, Ra, is the accumulator register. Ra is usually the source of transfers to memory or is the destination of transfers from memory. The second register, Rm, is typically a memory pointer. Rm may have uses other than a memory pointer. In single register instructions, Rm is sometimes used to expand one op-code into 16 instructions. UYK-20 Instruction Formats are shown in Figure 3.

(2) UYK-20 Register Format Field. Except for op-codes 40 through 47 (jump instructions), 60 through 63 (register immediate instructions), and 70 through 77 (I/O instructions), the two-bit register format field is used according to simple, well defined rules. These rules are:

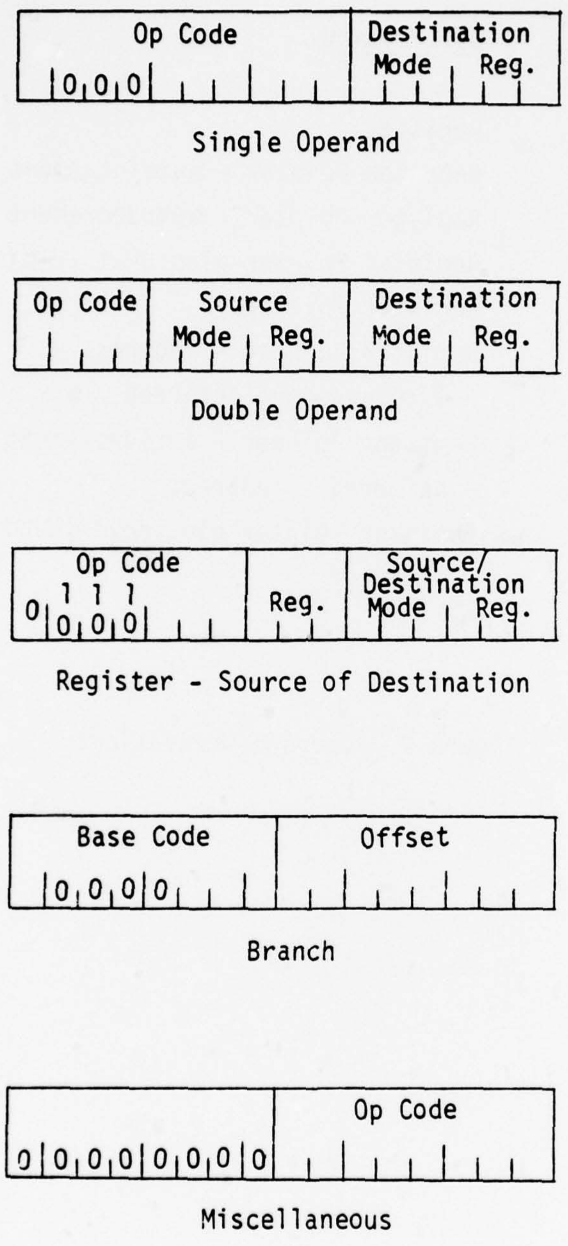

Figure 1. PDP-11 Instruction Formats

Mode	Description
0	Register
2	Register Pointer - autoincrement
4	Register Pointer - autodecrement
6	Register Pointer plus next 16-bit word
1	Register Pointer
3	Register Pointer - indirect - autoincrement deferred
5	Register Pointer - autodecrement deferred - indirect
7	Register Pointer plus next 16-bit word - indirect

Figure 2. Operand Derivations

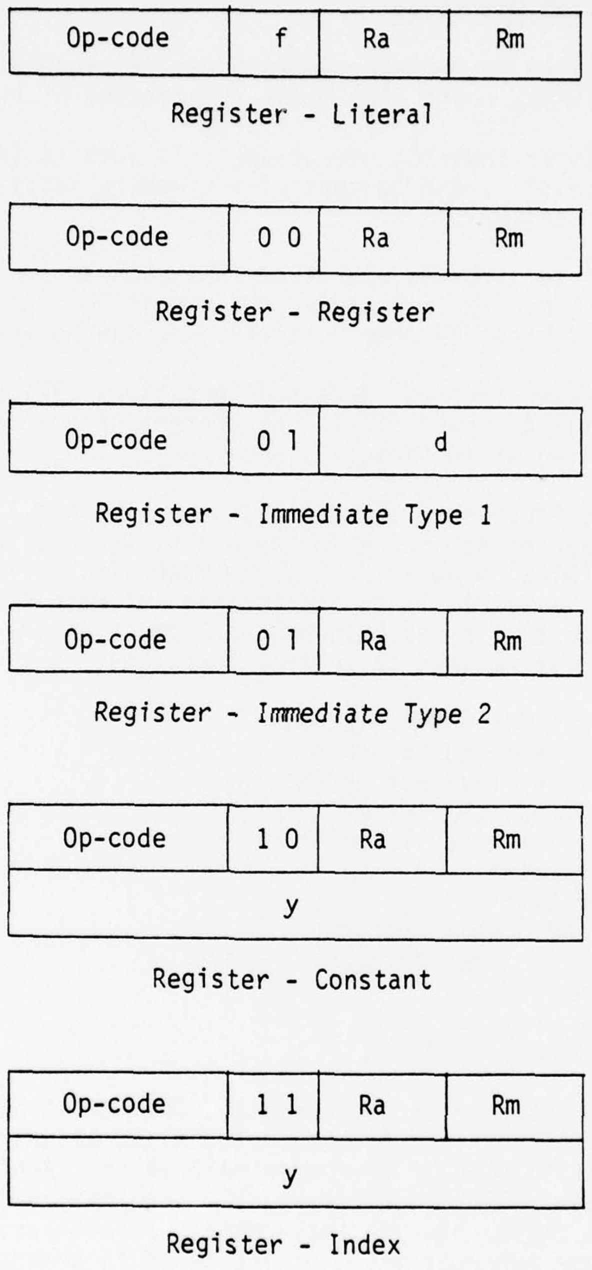

Figure 3. UYK-20, AYK-14 Instruction Formats

Format	Implied Operation
0	Register to Register Operations (RR) such as $(Ra) + (Rm) \rightarrow Ra$, where (Ra) means the content of register Ra.
1	Register indirect operations (RI) such as $(Ra) + (Y^*) \rightarrow Ra$ where Y^* is the content of the memory location with address (Rm).
2	Register indexed operation (RK) such as $(Ra) + Y \rightarrow Ra$ $Y = (Rm) + y$ if $Rm \neq Ro$ or $Y = y$ if $Rm = Ro$ and where y is the content of memory location following the instruction.
3	Register indexed, deferred operations (RX) such as $(Ra) + (Y) \rightarrow Ra$ where (Y) is the content of the memory location derived as in format 2, above.

c. AYK-15 (DAIS Processor) Characteristics. The AYK-15 is the avionics computer developed by the Air Force in support of the Digital Avionics Information System (DIAS) program. This computer also employs a 16-bit architecture and has a fairly large instruction set with several urique instructions in support of other DAIS hardware descriptions. Following is a list of AYK-15 features related to control functions.

1) 16 general purpose registers
2) Direct addressing to 65K
3) Single level indirect addressing
4) 16-bit immediate operand
5) Indexing via general registers
6) Floating point
7) Bit operation
8) 16-bit and 32-bit operands
9) 16 level vectored interrupt system
10) 2 internal timers
11) 1K block write protect
12) 16-bit memory word

(1) AYK-15 Instruction Description. The AYK-15 only employs two instruction formats (16-bit and 32-bit). The field boundaries of the 16-bit format are identical to that of the upper half of the 32-bit format. Figure 4 shows the two formats. GR1 and GR2 typically specify any of 16 general registers. The GR1 field, however, may contain a shift count, condition code, or bit number in some instructions. Rx is one of 15 general registers that can be used for indexing. The 16-bit address field is either a memory address or a 16-bit immediate operated for the instructions specifying immediate addressing.

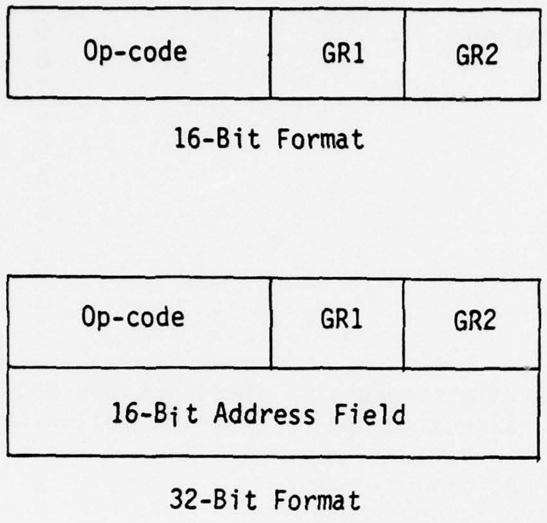

Figure 4. AYK-15 Instruction Formats

The 8-bit op-codes in the AYK-15 are denoted in hex and the first digit specifies the type of instruction:

1)	Bit operation	5
2)	Shift	6
3)	Jump	7
4)	Load	8
5)	Store	9
6)	Add	A
7)	Subtract	B
8)	Multiply	C
9)	Divide	D
10)	Logical	E
11)	Compare	F

The second hex digit generally specifies the modifier on the instruction type and the addressing option from the following set:

1)	Single precision direct	0
2)	Single precision regular to-regular	1
3)	Single precision indirect	2
4)	Single precision immediate	3
5)	Double precision direct	4
6)	Double precision regular-to-regular	5
7)	Double precision indirect	6
8)	Floating point direct	7
9)	Floating point regular-to-regular	8
10)	Floating point indirect	9

3. INDENTIFICATION OF CONTROL REQUIREMENTS

a. PDP-11 Instruction Decomposition. Figure 5 illustrates a controller decode of the PDP-11 instruction set. This is a general purpose scheme that is not optimized for speeding up register-to-register operations. Trade-offs between micro-memory size and execution speed could reduce the execution time of register-to-register operations at the expense of adding more micro instructions.

In this scheme once the new instruction has been fetched from memory and the program counter updated, an "instruction type" decode is performed on the upper eight bits of the instruction. There are four possible results: double operand, single operand, branch, and miscellaneous. The "instruction type" decode sorts an instruction into a category on the following basis:

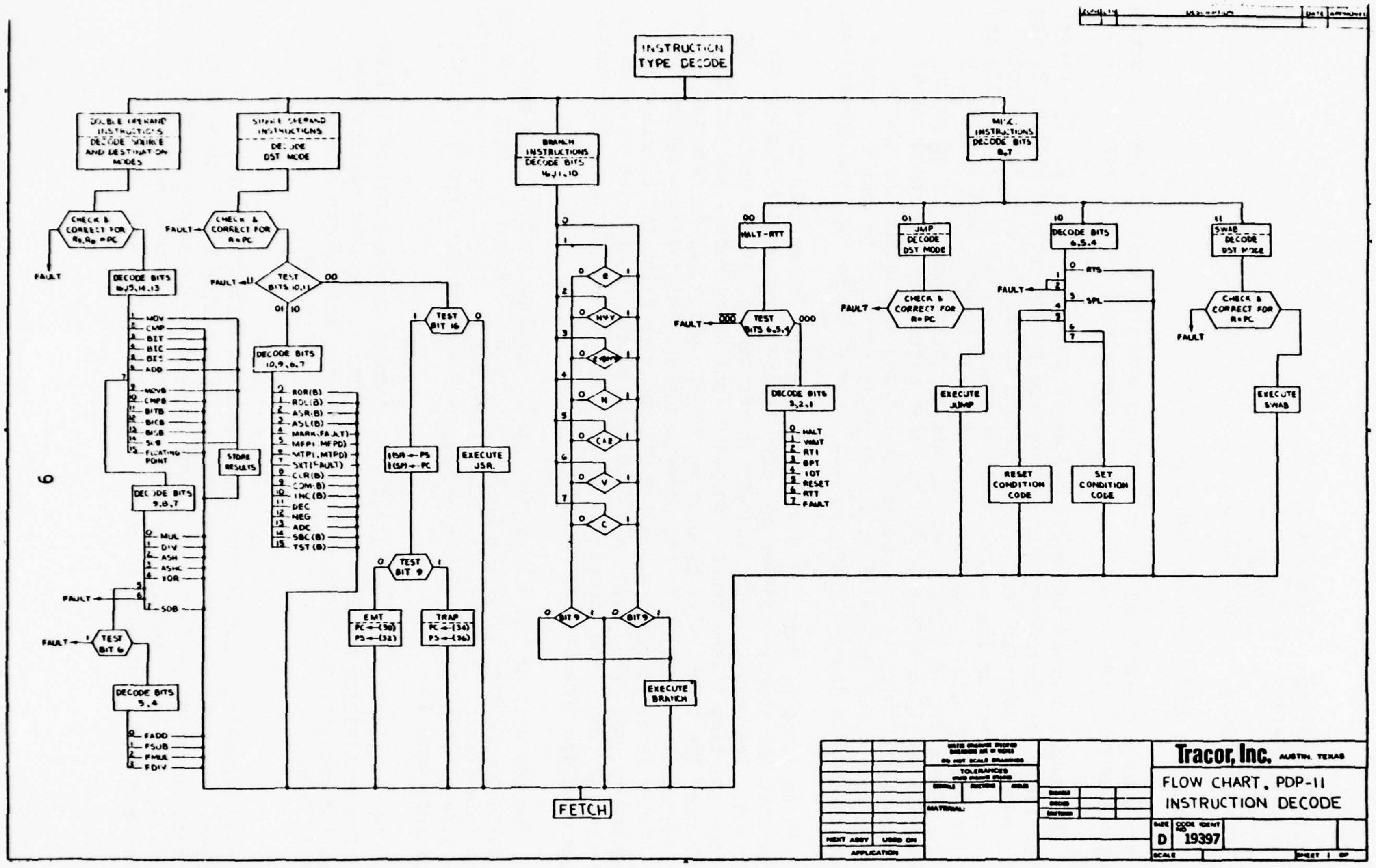

Figure 5. PDP-11 Instruction Decode Flow Chart

double operand	X NNN XXX X
single operand	X 000 XXX X
branch	X 000 0XX X
miscellaneous	0 000 000 0

where X = don't care and NNN ≠ not 000

Single the double operand set of instructions embodies most of the instruction decoding maneuvers, that set will serve to demonstrate the process.

The double operand instruction flow is shown in Figure 6. The first operation loads the source register, determined by instruction bits 9, 8, and 7, into temporary register Temp 6. The second operation loads Temp 6 into Temp 5 then branches on the source mode (instruction bits 12, 11, and 10) and whether or not the source register is the program counter (PC) (bits 9, 8, 7 = 111). There are 16 possible end points to the branch, four of which indicate that an illegal instruction has been detected. The remaining 12 branch end-points initiate the operations necessary to place the source operand in Temp 4 and the modified source register value in Temp 5. Temp 6 contains the source operand address which is only needed for byte operations.

As an example, Mode 7 with a source register other than the PC requires that the source register be added to the index word following the instruction. The result is the address of the address of the operand. The micro-code starting at the branch end-point indicating Mode 7 and not the PC first fetches the index value by placing the PC (which was incremented by two during instruction fetch) onto the address bus and initiating a memory fetch cycle. Prior to being restored in the GPU register file, the PC value is incremented by two. When available from memory, the index value is added to the content of Temp 6 and stored in Temp 6. Temp 6 then contains the source register plus the index value.

The addresss of the source operand is found by performing a memory fetch using the content of Temp 6 as an address and loading the data from that address into Temp 6. Finally, the source operand is determined by performing a memory fetch using Temp 6 as the address and loading the content of that address into Temp 4.

The next set of micro-instructions decodes the destination information. The destination operand is placed in Temp 1, the next value of the destination register is placed in Temp 2, and the destination address is placed in Temp 3.

Since many anomalies occur in the double operand instruction set, it is best to separate the byte and word instructions (instruction bit 16 true or false). The operation decode is first split into 16 paths by branching on instruction bits 16-13. The word instructions such as Move, Complement, Add and Subtract are then executed. The byte operations mask the source byte, aligned the

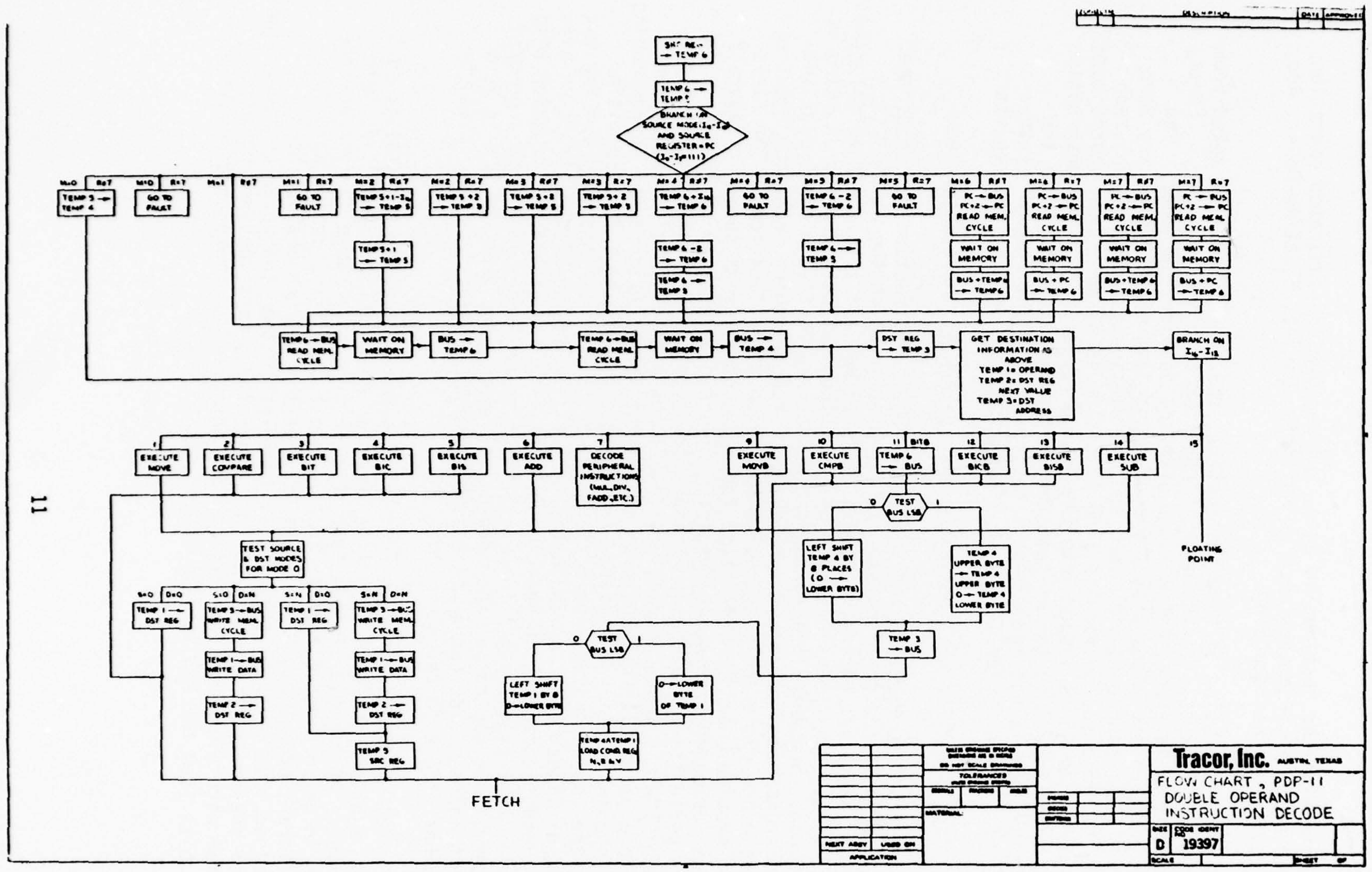
SRC REG → TEMP 6
TEMP 6 → TEMP 5
BRANCH ON SOURCE MODE: I₆-I₁₀ AND SOURCE REGISTER = PC (I₆-I₈ = 111)
GO TO FAULT
WAIT ON MEMORY
READ MEM. CYCLE
GET DESTINATION INFORMATION AS ABOVE TEMP 1 = OPERAND TEMP 2 = DST REG NEXT VALUE TEMP 3 = DST ADDRESS
BRANCH ON I₁₂-I₁₅
EXECUTE MOVE
EXECUTE COMPARE
EXECUTE BIT
EXECUTE BIC
EXECUTE BIS
EXECUTE ADD
DECODE PERIPHERAL INSTRUCTIONS (MUL, DIV, FADD, ETC.)
EXECUTE MOVB
EXECUTE CMPB
EXECUTE BICB
EXECUTE BISB
EXECUTE SUB
FLOATING POINT
TEST SOURCE & DST MODES FOR MODE 0
TEST BUS LSB
LEFT SHIFT TEMP 4 BY 8 PLACES (0 → LOWER BYTE)
TEMP 3 → BUS
FETCH
Tracor, Inc. AUSTIN, TEXAS
FLOW CHART, PDP-11 DOUBLE OPERAND INSTRUCTION DECODE
D 19397

Figure 6. PDP-11 Double Operand Instruction Decode Flow Chart

source byte with the destination byte, and perform the indicated operation. The peripheral instructions such as Multiply, Divide, Floating-Point, Add, etc., require further decoding.

After execution of some instruction, the results must be moved from the temporary registers to the general register file and/or memory. First, a four way branch is executed for the source and destination modes. The resulting branch indicates that the source and destination modes are zero (SD = 0,0), the source mode is zero and the destination mode is non-zero (SD = ON,). If a source mode is zero no action is taken. If a source mode is non-zero, Temp 5 is loaded into the source register. If the destination mode is zero, Temp 1 is loaded into the destination register. If the destination mode is not zero, Temp 1 is stored into the memory location determined by Temp 3, and Temp 2 is loaded into the destination register.

b. Controller Features Desirable for the PDP-11 Instruction Set.

(1) Field Masked Branching. The PDP-11 instruction decode scheme previously presented contains several examples in which the ability to determine that a set of instruction bits is either all zero, neither all zero nor all ones, or all ones is quite useful. The instruction group decoding is one example. Sequential field testing would require five micro-cycles and eight micro-memory locations. The field masked branching technique requires only one micro-cycle and consumes at most nine micro-locations. The field masking section under consideration for the controller is shown in Figure 7. The final proposed instruction register will probably be 10 or 12 bits wide in each controller to allow convenient field grouping. Each field mask register (FMR) is loaded during initialization with a set of ones and zeros. A one in a particular bit position causes the FMR to examine the corresponding bit in the instruction register. A zero in a particular FMR bit position causes the corresponding instruction register bit to be ignored. The FMR has two output lines; one line indicates that all of the enabled instruction bits are one and the other line indicates that all of the enabled instruction bits are zero. The eight ouput lines from the four FMR's are masked by a general register in the controller and merged with eight bits from the ROM to form a branch address. The number of possible branch destinations can, thus, be any number from 2 to 256.

(2) Field Branching. The PDP-11 also requires taking N bits, such as 3 or 4 bits from the instruction register and branching to 2^N possible locations. An example of this type of branch is the double operand op-code branch in which bits 16 through 13 are used to split the decode path into 15 different operations. The exact form of this branch has not been determined, but a likely protocol is to select either the upper or lower eight bits of the instruction register in a controller, mask the instruction register with a general register in the controller and merge the result with eight bits from the ROM to form the branch address.

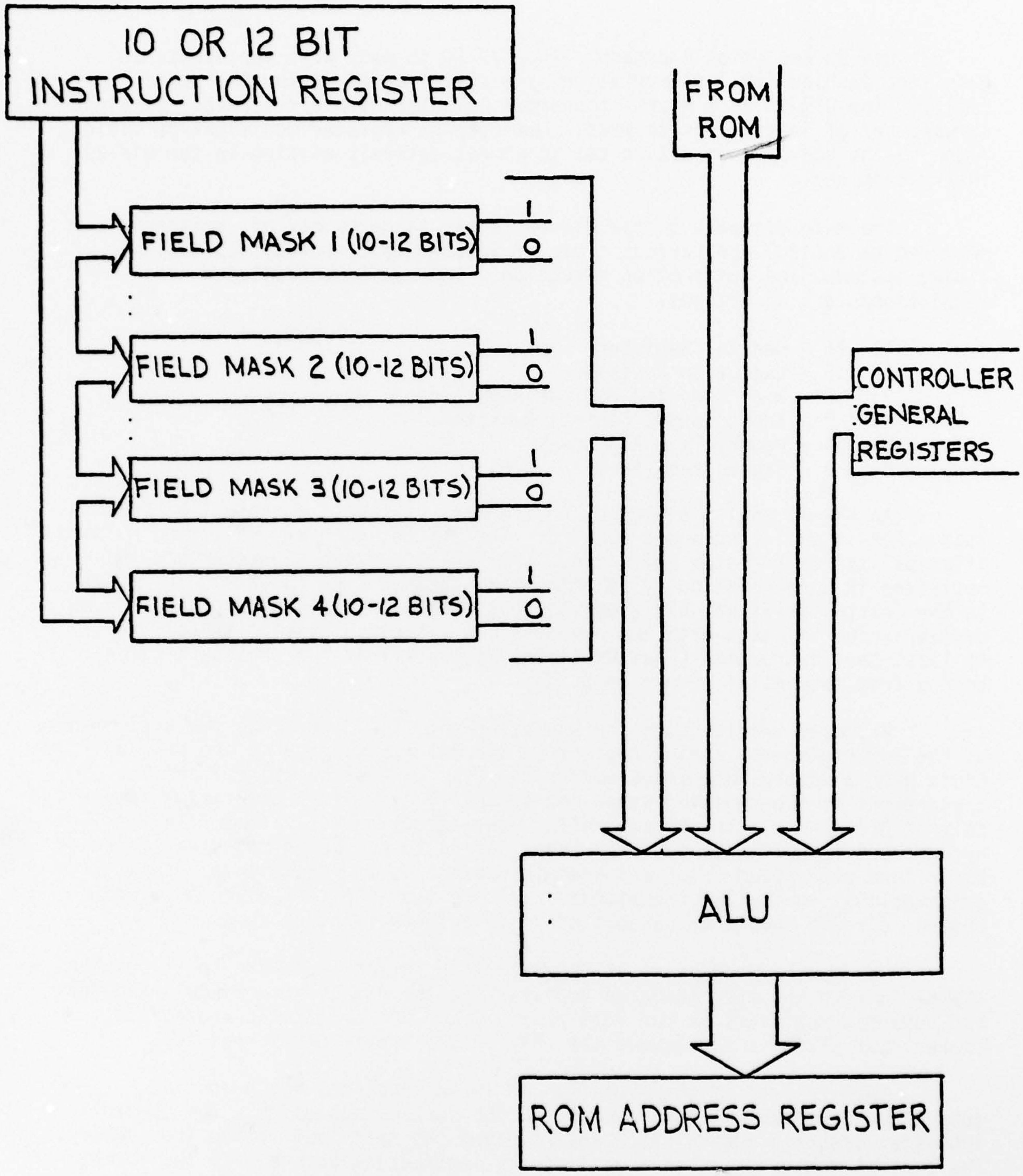
10 OR 12 BIT INSTRUCTION REGISTER
FROM ROM
FIELD MASK 1 (10-12 BITS)
FIELD MASK 2 (10-12 BITS)
FIELD MASK 3 (10-12 BITS)
FIELD MASK 4 (10-12 BITS)
1
0
CONTROLLER GENERAL REGISTERS
ALU
ROM ADDRESS REGISTER

Figure 7. Field Masking Section

c. UYK-20 Emulation Approach. The UYK-20 in many ways represents an excellent machine for implementation by a system based on the GPU parts family. The UYK-20 is a micro-programmed computer with a reasonably straightforward set of instruction formats. The complex register modification fields found in the PDP-11 instruction set is almost entirely missing in the UYK-20 instruction set.

The main drawback to the UYK-20 is the large number of registers required to control the various types of input/output operations, the two timing systems, and instruction execution. The estimated register requirements are as follows:

1) 16 - General Registers
2) 15 - Execution Registers
3) 6 - Real Time and Monitor Registers
4) 8 - Input/Output Control Registers
5) 64 - Memory Page Registers
6) 2 - Status Registers

An UYK-20 emulation would require most of the same controller features that a PDP-11 emulation would require. The UYK-20, however, will require more "flexibility" in register selection. The PDP-11 and the GPU reference two registers in concatenation by assuming that the register numbers only differ in the least significant bit (register even/register odd). The UYK-20 allows concatenation to start with any register and extend to sequential registers. At least one instruction (Load Multiple) allows all of the registers to be loaded from sequential memory locations.

Proposed architecture for emulating the UYK-20 would be the assignment of the I/O registers, timing registers, and program counter to one pair of GPU's with a mostly autonomous controller (the control subsystem) and the assignemnet of the register stack and temporary execution registers in two pair of GPU's with a second controller (the execution subsystem). This arrangement (see fig. 8) not only keeps logically connected registers together but allows overlapped fetch and execute operations and facilitates 32-bit arithmetic by controlled contatenation of the two 16-bit files. The micro-control circuit should be capable of handling both function types.

The general registers in the execution set of GPU's are split between the GPU's with the even numbered registers in the right pair of GPU's and the odd numbered registers in the left pair. In 32-bit arithmetic operations, either pair of GPU's can assume the role of the upper 16-bit register.

Near the end of the execution of an instruction, the execution subsystem would send a fetch command to the control subsystem. The control subsystem issues a memory fetch and receives the next instruction from memory. The control subsystem makes a preliminary instruction decode. If the format is RK or RX, the control subsystem fetches the next memory location and updates the program counter.

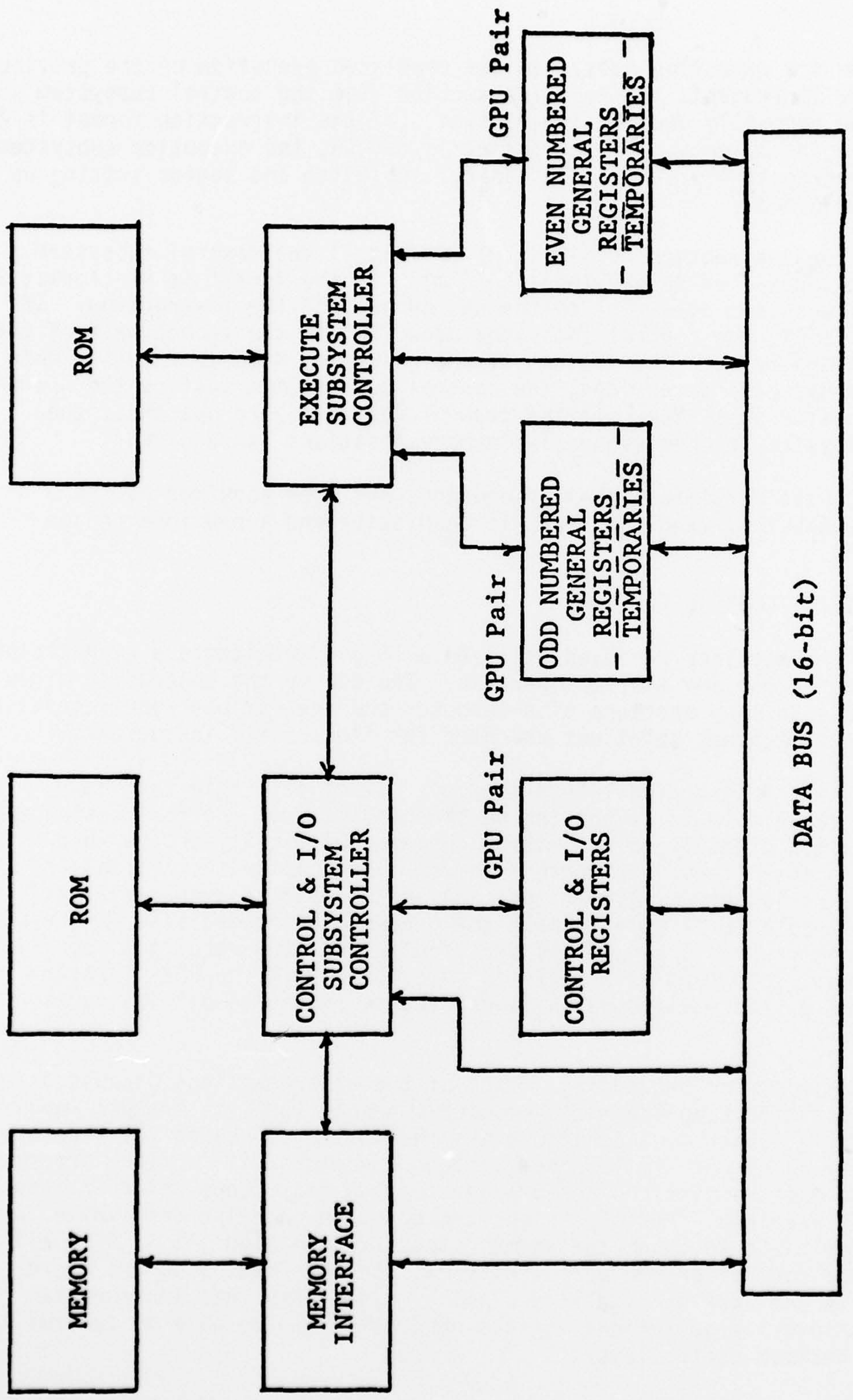
MEMORY
MEMORY INTERFACE
ROM
CONTROL & I/O SUBSYSTEM CONTROLLER
GPU Pair
CONTROL & I/O REGISTERS
ROM
EXECUTE SUBSYSTEM CONTROLLER
GPU Pair
ODD NUMBERED GENERAL REGISTERS
TEMPORARIES
GPU Pair
EVEN NUMBERED GENERAL REGISTERS
TEMPORARIES
DATA BUS (16-bit)

Figure 8. UYK-20 System Block Diagram

When the execution subsystem has completed execution of the previous instruction, it requests the next instruction from the control subsystem and receives the partially decoded instruction. If the instruction format is RR, execution begins at once. If the format is not RR, the execution subsystem transfers the content of Rm to the control subsystem and begins setting up the execution sequence.

If the instruction format is RI (format 1) the control subsystem fetches the content of memory location (Rm). If the format is RK (format 2), the control subsytem adds (Rm) to the second half of the instruction. If the format is RX, the control subsystem adds (Rm) to the second half of the instruction and fetches the content of the resulting memory address. Once the operand has been determined, the control subsystem transfers the operand to the execution subsystem. In the case of multiple word operands, the control subsystem fetches sequential memory locations as required.

Once the required number of operands has been acquired by the execution subsystem, execution runs to completion and a new instruction cycle begins.

4. SYNOPSIS OF CONTROL FUNTIONS

All of the machines reviewed utilized a 16-bit architecture with capablity for handling larger and smaller operands. The use of the controller circuit is applicable to many sections of a computer and the various requirements of the different machines point out the need for flexibility in the part.

a. Machine Instruction Interpretation. It is desirable for the controller to be able to select the pertinent bits from the instruction and map the information directly to the micro address. Difficulty occurs when patterns are established for certain fields in an instruction format and exceptions are inserted. All of the machines exhibit this characteristic. It is also desirable to be able to make the proper mapping decision in one micro-cycle and not require a sequential decision making flow requiring several micro-cycles. This type of problem is more obvious in the PDP-11 which incorporates multiple field definitions because of the 16-bit instruction limitation.

b. Micro-Command Sequencing. Many of the other sections discussed result in the output effecting the micro-command sequence (DMA, Interrupt, Input/Output, etc.). Other considerations are the desire to reduce the size of micro memory by having similar instructions sequenced with a micro program counter but most routines do not proceed through many steps before a branch decision is required. Most decisions are based on changing data value, which makes it desirable to input the normal status information (+, -, O/F, all zero) in the controller and efficiently map the information to the micro address. In the case of highly parallel architectures that incorporate parallel or overlap operations it is a requirement to be able to control the interface between controllers.

c. Memory Request. Memory systems in the newer generation processors have several independent sections to service and to resolve conflicts. Requests can exist from the CPU, I/O processor, DMA, and possible other areas depending on design. The memory may also incorporate interleaved banks to increase effective speed. In these cases control is required to administer priority without eliminating a section from having any access and to determine when requests are in conflict. Machines desiring interface to a variety of memory systems can also be easily accomodated with a micro controlled interace.

d. Direct Memory Access (DMA). A DMA channel requires control handshaking, maintaining the word count, addressing memory for data, possible serialization/parallelization, format checks, and memory interface. All of these should be readily handled by the controller and GPU's depending on word size.

e. Register Designation. Selection of the various registers in the GPU ~as to be available to the micro-programmer as well as being designated in ~e instruction. The ability to test an instruction register field for all ~eros is required by all machines to determine indexing requirments. The ability to increment the original register select value and compare to another value is required for load and store multiple instructions.

f. Interrupts. Multiple vectored interrupts, maskable under program control, with assigned priority will probably be handled by a custom designed circuit, but the output will be an input to the control sequence. Multiple leveled interrupt systems such as the UYK-20 could use the controller to merge to various levels.

g. User Interface. Data registers and machine status in an LSI machine are not available for monitoring except under micro-program control. The controller should have the ability to micro-cycle and output information to the operator as would be the case in previous consoles.

h. Conditional Status Inputs/Storage. ALU status information (+, -, O/F, all zero) is required to be compared or mapped in support of conditional brand instructions. These values also make up part of a status register in each machine that must be stored and loaded when servicing interrupts. Since this information is required by the controller anyway, it is desirable that status be stored in the controller.

i. Input/Output. The I/O processor in the UYK-20 is similiar in operation to a CPU and would therefore have similiar requirments for the controller. In addition, the controller could be used to handle channel protocol and control GPU's when data manipulation is required.

5. RECOMMENDATION

The identification of control functions relates tothe control design as a list of features to be considered in its design. Following is a list of recommendations based on control requirements of the machines surveyed:

1) Register for storage of portion of instruction pertinent to controller
2) Program counter for in-line routines
3) Temporary storage registers for subroutine linkage
4) Iteration counter for sequential execution of the same micro instruction
5) Discrete inputs for micro-control branching
6) Mapping and input comparisons for multiply and divide data dependent decisions
7) Register for storage of ALU status and ability to output
8) Mask registers to obtain field classifications of the instruction
9) Mask registers for mapping instruction field to micro address
10) Ability to control interface between controllers operating simultaneously

SECTION II

1. GENERAL PROCESSOR UNIT (GPU) SUPPORT

Extensive testing and evaluation of the GPU circuit has been accomplished under separate Air Force contracts. Tracor has supported this activity and submitted solutions to the items identified. Two of these recommendations effected control definitions.

Make the circuit output identified by the "Destination Control" availalbe to output for all destination selections if desired. To accomplish this a tri-state control was requested in place of the "All zero detect input." This resulted in a slight redesign of the all-zero detect circuit making it a one pin pull down implementation. Also, a no load state was added.

A redistribution of the terms time shared on the most significant connect pins was recommended to allow isolation of the individual GPU circuits in a system and to provide a complete arithmetic status at one time.

An update of the control definitions is provided in Tables I through V. Table VI reflects the change in pin assignment and Figure 9 shows the change to the block diagram resulting from Item 1.

Table I. Source Select Control

Reference	Inputs S1	S0	Port 1 Source	Port 2 Source	Condition/ Description	
SS0	0	0	(R)	(T)	$\overline{\text{AD1}}$	
			(R)	(P2B)	AD1	
SS1	0	1	DI, R	(T)	$\overline{\text{AD1}}$	Enable R if LOAD required
			DI, R	(P2B)	ADI	
SS2	1	0	(R)	DI	$\overline{\text{AD1}}$	
			(R)	(T)	AD1	
SS3	1	1	(R + 1)	DI	$\overline{\text{AD1}}$	
			(R + 1)	(T)	AD1	

Table II. Data Type Selector Control

Reference	Inputs D2	D1	D0	Port 1 ALC In	Port 2 ALC In	Condition/ Description
DS0	0	0	0	Zero	False	
DS1	0	0	1	True	False	
DS2	0	1	0	P1B/2	False	AD1
DS2	0	1	0	False	False	$\overline{\text{AD1}}$
DS3	0	1	1	False	Zero	
DS4	1	0	0	Zero	True	
DS5	1	0	1	True	True	
DS6	1	1	0	P1B/2	True	AD1
DS6	1	1	0	False	True	$\overline{\text{AD1}}$
DS7	1	1	1	True	Zero	

Table III. ALC Control

Reference	Inputs A1	Inputs A0	Function	Comments
AD0	0	0	ADD	First port 2 source, external carry-in
AD1	0	1	ADD	Second port 2, source external carry-in
AD2	1	0	AND	First port 2 source, logical "1" carry-in*
AD3	1	1	OR	First port source, logical "0" carry-in*

*Group level propogate of external carry-in

Table IV. Destination Select Control

Reference	Inputs M2	M1	M0	Description
A00	0	0	0	Direct store input to register file*
A01	0	0	1	ALC result left shift one into port 1*
A02	0	1	0	ALC result right shift one into port 1*
A03	0	1	1	ALC result right shift two into port 1*
A04	1	0	0	No load*
A05	1	0	1	ALC result no shift into port 1*
A06	1	1	0	P2B to circuit output, ALC result No shift into port 1 if load clock
A07	1	1	1	P1B to circuit output, ALC result No shift into port 1 if load clock

*ALC result to circuit output

Table V. Boundary and Connect Control

Reference	Inputs			Description	MS Connection Outputs	
	C2	C1	C0		MXH(0)	MXH(1)
RF0	0	0	0	No connection	Hi Z	Hi Z
RF1	0	0	1	Connection for normal intra-circuit shift operations	$(\text{ALC Out})_7$	Hi Z
RF2	0	1	0	Logical "1" shift in (MSB if right shift, LSB if left shift)	$(\text{ALC Out})_7$	Overflow
RF3	0	1	1	Logical "0" shift in (MSB if right shift, LSB if left shift)	$(\text{ALC Out})_7$	Overflow
RF4	1	0	0	MSB extend for right shift only - complimented if overflow	$(\text{ALC Out})_7$	$(\text{ALC Out})_6$
RF5	1	0	1	Remainder Correction	$(\text{ALC Out})_7$	TS
RF6	1	1	0	Divide Code	$(\text{ALC Out})_7$	$(\text{P2B})_7$
RF7	1	1	1	$(\text{ALC Out})_7$ TS; Divide Code	$(\text{ALC Out})_7$	$(\text{P2B})_7$

TS - Temporary Storage

Table VI. Pinouts of the GPU

Pin No.	Function	Pin No.	Function
1	V_{SS}(-V)	25	VDD (+V)
2	Data out (B4)	26	R2
3	Data out (B5)	27	R3
4	Data out (B6)	28	T2
5	Data out (B7)	29	T3
6	Data out (B8)	30	R1
7	Carry-out	31	T1
8	Tri-State Enable	32	T0
9	AZD out	33	R0
10	MX H (0)	34	M1
11	MX H (1)	35	M0
12	C0	36	M2
13	C1	37	MXL (0)
14	C2	38	MXL (1)
15	Data in (B8)	39	LC
16	Data in (B7)	40	D2
17	Data in (B6)	41	A0
18	Data in (B5)	42	A1
19	Data in (B4)	43	Carry-in (CRI)
20	Data in (B3)	44	D1
21	Data in (B2)	45	D0
22	Data in (B1)	46	Data out (B1)
23	S1	47	Data out (B2)
24	S0	48	Data out (B3)

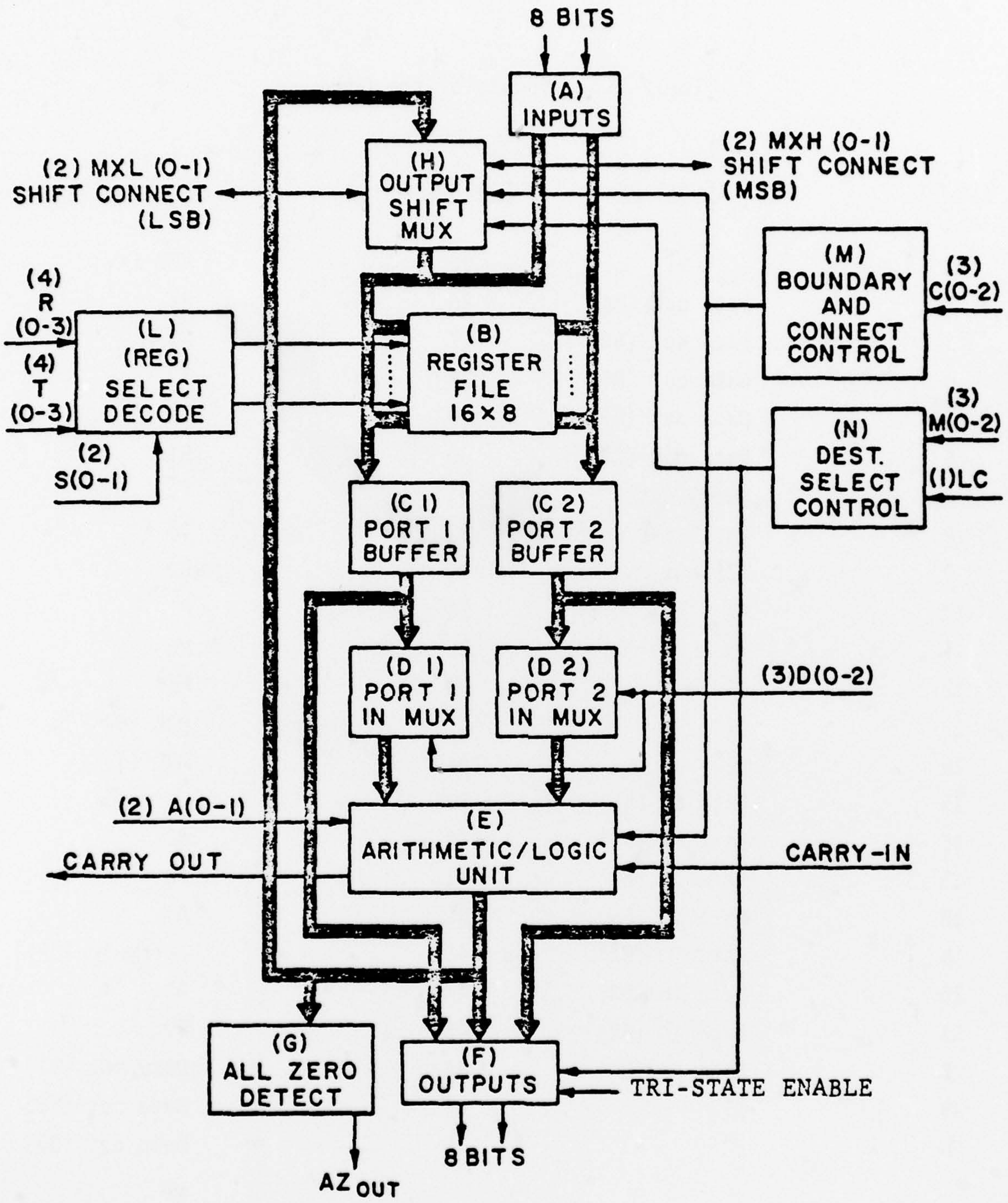

Figure 9. GPU organization

☆U.S.Government Printing Office: 1978 — 657-002/207